

Building An Ad-Hoc Windows Cluster for Scientific Computing

By
Andreas Zimmerer

Submitted in partial fulfillment
of the requirements for the degree of
Masters of Science
in Computer Science

at

Seidenberg School of Computer Science
and Information Systems

Pace University
November 18, 2006

We hereby certify that this dissertation, submitted by Andreas Zimmerer, satisfies the dissertation requirements for the degree of *Doctor of Professional Studies in Computing* and has been approved. _____

Name of Thesis Supervisor

Date

Chairperson of Dissertation Committee _____

Name of Committee Member 1

Date

Dissertation Committee Member _____

Name of Committee Member 2

Date

Dissertation Committee Member _____

Seidenberg School of Computer Science
and Information Systems
Pace University 2006

Abstract

Building An Ad-Hoc Windows Cluster for Scientific Computing

by

Andreas Zimmerer

Submitted in partial fulfillment
of the requirements for the degree of
M.S. in Computer Science
September 2006

Building an Ad-Hoc Windows Computer Cluster is an inexpensive way to perform scientific computing. This thesis describes how to build a cluster system out of common Windows computers and how to perform chemical calculations. It gives an introduction to software for chemical high performance computing and discusses several performance experiments. These experiments show how the relationship between topography, network connections, computer hardware and number of nodes effect the performance of the computer cluster.

Contents

1	Introduction	1
1.1	Preamble	1
1.2	Structure	1
2	Grid and Cluster Computing	3
2.1	Outline of the Chapter	3
2.2	Introduction to Cluster and Grid Concepts	3
2.3	Definitions Grid	4
2.3.1	Ian Foster's Grid Definition	4
2.3.2	IBM's Grid Definition	4
2.3.3	CERN's Grid Definition	5
2.4	Definitions of Cluster	5
2.4.1	Robert W. Lucke's Cluster Definition	5
2.5	Differences between Grid and Cluster Computing	6
2.6	Shared Memory VS Message Passing	6
2.6.1	Message Passing	6
2.6.2	Shared Memory	7
2.7	Benchmarks	7
2.8	The LINPACK Benchmark	8
2.9	The future of Grid and Cluster Computing	8
3	WMPI	10
3.1	Outline of the Chapter	10
3.2	Introduction to WMPI	10
3.2.1	MPI: The Message Passing Interface	10
3.2.2	WMPI: The Windows Message Passing Interface	10
3.3	Internal Architecture	11
3.3.1	The Architecture of MPICH	11
3.3.2	XDR: External Data Representation Standard	11
3.3.3	Communication on one node	12
3.3.4	Communication between nodes	12
3.4	The Procgroup File	12
4	PC GAMESS	14
4.1	Introduction to PC GAMESS	14
4.2	Running PC GAMESS	15
5	NAMD	16
5.1	Introduction to NAMD	16
5.2	Running NAMD	16

6	The Pace Cluster	18
6.1	Overview	18
6.2	Adding a Node to the Pace Cluster	18
6.2.1	Required Files	18
6.2.2	Creating a New User Account	19
6.2.3	Install WMPI 1.3	19
6.2.4	Install PC GAMESS	20
6.2.5	Install NAMD	20
6.2.6	Firewall Settings	21
6.2.7	Check the Services	23
6.3	Diagram: Runtimes / Processors	24
6.4	Diagram: Number of Basis Functions / CPU Utilization	28
6.5	Network Topology and Performance	29
6.6	Windows VS Linux	40
6.7	Conclusion of the Experiments	44
6.8	Future Plans of the Pace Cluster	45
7	The PC GAMESS Manager	46
7.1	Introduction to the PC GAMESS Manager	46
7.2	The PC GAMESS Manager User's Manual	46
7.2.1	Installation	46
7.2.2	The First Steps	47
7.2.3	Building a Config File	49
7.2.4	Building a NAMD Nodelist File	50
7.2.5	Building a Batch File	50
7.2.6	Run the Batch File	51
7.2.7	Save Log File	52
7.3	RUNpcg	52
7.4	WebMo	54
7.5	RUNpcg, WebMo and the PC GAMESS Manager	55
8	Conclusion	57
A	Node List of the Pace Cluster	59
A.1	Cam Lab	59
A.2	Tutor Lab	59
A.3	Computer Lab - Room B	59
B	PC GAMESS Inputfiles	62
B.1	Phenol	62
B.2	db7	62
B.3	db6 mp2	63
B.4	db5	64
B.5	Anthracene	65

B.6	18cron6	66
-----	-------------------	----

1 Introduction

1.1 Preamble

In the early days of computers, high performance computing was very expensive. Computers were not as common as now-a-days and supercomputers had only a fraction of the computing power and memory an office computer has today. The fact that supercomputers are very expensive did not change over the decades, high performance computers still cost millions of dollars. Today however, office computers are more widely used and have become more powerful over the last years, which has opened a completely new way of creating inexpensive high performance computing. The idea of an ad-Hoc Microsoft Windows Cluster is to combine the computing power of common Windows office computers. Institutions like universities, companies or government facilities usually have many computers which are not used during the night or holidays, and the computing power of these machines can be used for a cluster. This thesis demonstrates how it is possible to build a high performance cluster with readily available hardware combined with free available software.

1.2 Structure

An introduction to grid and cluster computing is given in the next chapter. It will define grid and cluster computers as well as point out the differences between them. The message passing model will be compared with the shared memory model, followed by a short introduction to benchmarks. The third chapter will discuss WMPI, the technology used for communication between the computers in the cluster built as part of this thesis- the Pace Cluster. Chapter four is about PC GAMESS and chapter five is about NAMD, two programs used to perform high performance chemical computations with the cluster. Chapter six discusses the Pace Cluster. It

describes the physical topology of the computers it consists of and explains how to add new nodes. The result of different runs are discussed and compared to runs of a Linux cluster. The chapter closes with a future outlook of the Pace cluster. The following chapter introduces the PC GAMESS Manager, a user friendly tool which was developed as part of the thesis to create config files and to start PC GAMESS runs. It will also be compared to similar software tools. The thesis closes with a conclusion.

2 Grid and Cluster Computing

2.1 Outline of the Chapter

This chapter will present an introduction to cluster and grid concepts. The first point discusses the basic idea of a grid or cluster systems and the purposes for which they are built. Following this point are several varying professional definitions of the terms grid and cluster, indicating the differences between these concepts. The chapter ends with a future outlook of grid and cluster computing.

2.2 Introduction to Cluster and Grid Concepts

Clusters as well as grids consist of a group of computers, which are coupled together to perform high-performance computing. Grids and clusters built from low-end servers are very popular because of the low costs compared to the cost of large supercomputers. These low cost clusters are not able to do very high-performance computing, but the performance is in most cases sufficient. Applications of grid and cluster systems include calculations for biology, chemistry and physics, as well as complex simulation models used in weather forecasting. Automotive and aerospace applications use grid computing for collaborative design and data-intensive testing. Financial services also use clusters or grids to run long and complex scenarios. An example of a high-end cluster is the *Lightning* [1] at Opteron Supercomputer Cluster which runs under Linux. It consists of 1408 dual-processor Opteron servers and can deliver theoretical peak performance of 11.26 trillion floating-point operations per second (11.26 terra FLOPS). It works for Los Alamos National Laboratory's [2] nuclear weapons testing program and simulates nuclear explosions. It is worth over \$10 million.

The World Community Grid [3], a project at IBM, is an example of one famous grid.

Consisting of thousands of common PCs from all over the world, it establishes the computing power that allows researchers to work on complex projects like human protein folding or identifying candidate drugs that have the right shape and chemical characteristics that block HIV protease. Once the software is installed and detects that the CPU is idle, it requests data from a Word Community Grid server and performs a computation.

2.3 Definitions Grid

There are many different definitions for a grid. The following are the most important.

2.3.1 Ian Foster's Grid Definition

Ian Foster[4] is known as one of the big grid experts in the world. He created the Distributed Systems Lab at the Argonne National Laboratory, which has pioneered key grid concepts, developed Globus software (the most widely deployed grid software), and he led the development of successful grid applications across the sciences. According to Foster, a grid has to fulfill three requirements:

1. The administration of the resources is not centralized
2. Protocols and interfaces are open.
3. A grid delivers various qualities of services to meet complex user demands.

2.3.2 IBM's Grid Definition

IBM defines a grid as the following[6]:

Grid is the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and

a vast array of other computing resources over the Internet. A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across multiple administrative domains based on the resources availability, capacity, performance, cost and user's quality-of-service requirements.

2.3.3 CERN's Grid Definition

CERN , the European Organization for Nuclear Research, has the world's largest particle physics laboratory. CERN researchers use grid computing for their calculations. They define a grid as[7]:

A Grid is a service for sharing computer power and data storage capacity over the Internet. The Grid goes well beyond simple communication between computers, and aims ultimately to turn the global network of computers into one vast computational resource.

2.4 Definitions of Cluster

2.4.1 Robert W. Lucke's Cluster Definition

Robert W. Lucke, who worked on one of the world's largest Linux clusters at Pacific Northwest National Laboratories, defines the term cluster in his book "Building Clustered Linux Systems" as following[8]:

A closely coupled, scalable collection of interconnected computer systems, sharing common hardware and software infrastructure, providing

a parallel set of resources to services or applications for improved performance, throughput, or availability.

2.5 Differences between Grid and Cluster Computing

The terms grid and cluster computing are often confused and both concepts are very closely related. One major difference is that a cluster is a single set of nodes, which usually sits in one physical location, while a grid can be composed of many clusters and other kinds of resources. Grids can occur in different sizes from departmental grids over enterprise grids to global grids. Clusters share data and have a centralized control. The trust level between grids is lower than in a cluster system, because grids are more loosely tied than clusters. Hence they don't share memory and have no centralized control. A grid is more a tool for optimized workload, that shares independent jobs. A computer receives a job and calculates the result. Once the job is finished the node returns the result and performs the next job. The intermediate result of a job does not affect the other calculations, which run in parallel at the same time, so there is no need for an interaction between jobs. But there may exist resources like storage, which is shared by all nodes.

2.6 Shared Memory VS Message Passing

For parallel computing in a cluster there are two basic concepts for jobs to communicate with each other: the message passing model and the virtual shared memory model. [9]

2.6.1 Message Passing

In the message passing model each process can only access its memory. The processes send messages to each other to exchange data. MPI(message passing interface) is one

realization of this concept. The MPI library consist of routines for message passing and was designed for high performance computing. A disadvantage of this concept is that a lot of effort is required to implement MPI code as well as maintaining and debugging it. PC GAMESS, the quantum chemistry software discussed in this thesis, uses this approach and the Pace Windows Cluster works with WMPI(Windows Message Passing Interface).

2.6.2 Shared Memory

The Virtual Shared Memory Model is sometimes termed as Distributed Shared Memory Model or Partitioned Global Address Space Model. The idea of the Shared Memory Model is to hide the message passing commands from the programmer. Processes can access the data items shared across distributed resources and this data is then used for the communication. The advantages to the Shared Memory Model are that it is much easier to implement than the Message Passing Model and it costs much less to debug and to maintain the code. The disadvantage is that the high level abstraction costs in performance and is usually not used in classical high performance applications.

2.7 Benchmarks

Benchmarks are computer programs used to measure performance. There are different kinds of benchmarks. Some measure the CPU power with floating point operations, others draw moving 3D objects to measure the performance of 3D graphic cards or run against compilers. There are also benchmarks to measure the performance of database systems.

2.8 The LINPACK Benchmark

The LINPACK benchmark is often used to measure the performance of a computer cluster. It was first introduced by Jack Dongarra and is based on LINPACK [10], a mathematical library. It measures the speed of a computer solving n by n matrices of linear equations. The program uses the Gaussian elimination with partial pivoting. To solve an n by n system there are $\frac{2}{3} * n^3 + n^2$ floating point operations necessary. The result is measured in flop/s (floating point operations per second). HPL (High-Performance LINPACK Benchmark) is a variant of the LINPACK Benchmark used for large-scale distributed-memory systems. The TOP500 list [11] of the fastest supercomputers all over the world uses this benchmark to measure the performance. It runs with different matrix sizes n to search the matrix size where the best performance is achieved. The number 1 position in the TOP500 is the BlueGene/L System. It was developed by IBM and National Nuclear Security Administration (NNSA). It reached the LINPACK Benchmark of 260.6 TFlop/s (teraflops). BlueGene is the only system that runs over 100 TFlop/s.

2.9 The future of Grid and Cluster Computing

The use of Grid Computing is on the rise. [12] IBM called grid computing “the next big thing” and furnished their new version of WebSphere Application Server with grid-computing capabilities. IBM wants to bring grid capabilities to the commercial customers and to enable them to balance web server workloads in a much more dynamic way. Microsystems wants to offer a network where one can buy computing time. [13] Even Sony has made the move toward grid computing in its grid-enabled Play Station 3 [14]. Other game developers, especially online publishers and infrastructure providers for massively multilayer PC games, focused on grid computing as well. Over the last decade clusters of common PCs have become an

inexpensive form of computing. Cluster architecture has also become more sophisticated. According to Moore's law[15], the performance of the clusters will continue to grow as the performance of the CPUs grows, as well as storage capacity grows and system software improves. The new 64 Bit processors could have an impact especially on low-end PC clusters. Other new technologies could have an impact on the future performance of clusters as well, such as better network performance through optical switching, 10 Gb Ethernet or Infiniband.

3 WMPI

3.1 Outline of the Chapter

This chapter will give an introduction to the concepts of WMPI. First an understanding of WMPI is given as well as its usage. Following is a description of the architecture and how WMPI works internally. Finally the procgroup file is described and its usage is explained.

3.2 Introduction to WMPI

3.2.1 MPI: The Message Passing Interface

The Message Passing Interface (MPI) [16] provides standard libraries for compiling programs. MPI processes on different machines, in a distributed memory system, communicate using messages. Using MPI is a way to turn serial applications into parallel ones. MPI is typically used in cluster computing to facilitate communication between nodes. The MPI standard was developed by the MPI Forum in 1994.

3.2.2 WMPI: The Windows Message Passing Interface

WMPI (Windows Message Passing Interface) is an implementation of MPI. The Pace Cluster uses WMPI 1.3, which is not the latest version, but a free one. WMPI was originally free but became a commercial product with WMPI II [17]. WMPI implements MPI for the Microsoft Win32 platform and is based on MPICH 1.1.2. WMPI is compatible with Linux and Unix workstations, and it is possible to have a heterogeneous network of Windows and Linux/Unix machines.

WMPI 1.3 comes with a daemon that runs on every machine. The daemon receives and sends MPI messages and is responsible for smooth communication between

the nodes. High speed connections like 10 Gbps Ethernet [18], Infiniband[19] or Myrinet[20] are supported. WMPI 1.3 can be used with C, C++ and FORTRAN compilers. It also comes with some cluster resource management and analysis tools. One reason that WMPI is so popular is the fact that Win32 platforms are widely available and the increased performance of single workstations.

3.3 Internal Architecture

3.3.1 The Architecture of MPICH

MPICH, runs on many Unix systems, was developed by the Argonne National Laboratory and the Mississippi State University. The designers of WMPI [21] wanted a solution that is compatible with Linux/Unix so they considered an MPICH compatible WMPI implementation as the fastest and most effective way. The architecture of MPICH consists of independent layers. MPI functions are handled by the top layer and the underlying layer works with an ADI (Abstract Device Interface). The ADI has the purpose of handling different hardware-specific communication subsystems. One of these subsystems is the p4, a portable message passing system, which is used for UNIX systems communication over TCP/IP. P4 is an earlier project of the Argonne National Laboratory and the Mississippi State University.

3.3.2 XDR: External Data Representation Standard

It is not necessary that all nodes have the same internal data representation. WMPI uses XDR (External Data Representation Standard)[22] for communication between two systems with different data representation. XDR is a standard to describe and encode data. The conversion of the data to the destination format is transparent to the user. The language itself is similar to the C programming language, however it can be only used to describe data. According to the standard it is assumed that

a byte is defined as 8 bits of data. The hardware encodes and sends the data in a way that the receiver hardware decodes it without loss of information.

WMPI has only implemented a subset of XDR and uses it only when absolutely necessary.

3.3.3 Communication on one node

Processes on the same machine communicate via shared memory. Every process has its own distinct virtual address space, but the Win32 API provides mechanisms for resource and memory sharing.

3.3.4 Communication between nodes

Nodes communicate over the network using TCP. To access TCP, a process uses Win Sockets. Win Sockets is a specification that defines how Windows network software should access network services. Every process has a thread, which receives the incoming TCP messages and puts them in a message queue. This all happens transparently in WMPI, which must check only the message queue for incoming data.

3.4 The Procgroupp File

The first process of a WMPI program is called the big master. It starts the other processes, which are called slaves. The names or IP addresses of the slaves are specified in the procgroupp file. The following is an example procgroupp file:

```
local 0 pace-cam-02 1 C:\PCG\pcgamess.exe
pace-cam-12 2 C:\PCG\pcgamess.exe
172.168.1.3 1 C:\PCG\pcgamess.exe
```

The 0 in first line indicates how many additional processes are started on the local machine, where the big master is running. Local 1 would indicate a two CPU machine and that another process has to be started. For every additional node a line is added. The line begins with the Windows hostname or the IP address, followed by a number indicating how many CPUs the machine has. The path specifies the location of the WMPI program that should be run.

4 PC GAMESS

4.1 Introduction to PC GAMESS

PC Gamess [23], an extension of the GAMESS(US)[24] program, is a DFT(Density Functional Theory) computational chemistry program, which runs on Intel-compatible x86, AMD64, and EM64T processors and runs parallel on SMP systems [25] and clusters. PC GAMESS is available for the Windows and the Linux operating systems. Dr. Alex A. Granovsky coordinates the PC GAMESS project at the Moscow State University in the Laboratory of Chemical Cybernetics. The free GAMESS(US) version was modified to extend its functionality and the Russian researchers replaced 60-70% of the original code with a more efficient one. They implemented DFT and TDDFT(Time Dependent DFT) as well as algorithms for 2-e integral evaluation for direct calculation method. Other features are efficient MP2(Mollder-Plesset electron correlation) energy and gradient modules as well as very fast RHF(Restricted Hartree Fock) MP3/MP4 energy code. Another important factor that makes PC GAMESS high-performance is the usage of efficient libraries on assembler-level. Additional to the libraries from the vendors like Intel's MKL(Math Kernel Library), the researchers in the Laboratory of Chemical Cybernetics of the Moscow State University wrote libraries themselves. Dr. Alex A. Granovsky's team used different FORTRAN and C compilers, like the Intel vv. 6.0-9.0 or the FORTRAN 77 compiler v. 11.0 , to compile the source code of PC GAMESS. The GAMESS(US) version is frequently updated and the researchers at the Moscow State University adopt the newest features.

4.2 Running PC GAMESS

Initially one has to create a procgroup file, like described in the chapter WMPI. This file has to be in the directory C:\PCG\ and must have the ending *.pg*. To select the input file one must open the command prompt and set the variable input to the wanted path. For example:

```
set input=C:\PCG\samples\BENCH01.INP
```

Then run the PC GAMESS executable and enter the working directory, followed by the location of the output file as parameter. For example:

```
c:\PCG\pcgame.exe c:\pcg\work → C:\PCG\samples\BENCH01.out
```

5 NAMD

5.1 Introduction to NAMD

NAMD[26] is a parallel code for simulation of large biomolecular system and was designed for high-performance by the Theoretical Biophysics Group at the University of Illinois. NAMD is free for non-commercial use and can be downloaded after completing an online registration at the NAMD web site ¹.

5.2 Running NAMD

In order to run NAMD it is necessary to create a nodelist file, which contains the Windows hostnames or IP addresses of the nodes. The nodelist file is initiated by the word *group main*. An example would be:

```
group main host pace-cam-01
host pace-cam-02
host pace-cam-03
host pace-cam-04
host 172.20.102.62
host 172.20.102.214
host 172.20.103.119
host 172.20.103.112
```

NAMD is started by the Charm processes. This is done by giving Charm: the path to the NAMD executable, the number of processors it should be run on, the path

¹<http://www.ks.uiuc.edu/Research/namd/>

to the nodelist file, and to the NAMD input file. An example would be:

```
c:\NAMD\charmrun.exe c:\NAMD\namd2.exe +p2 ++nodelist c:\namd\apoa1\namd.nodelist  
c:\namd\apoa1\apoa1.namd
```

The number of processors is indicated by `+pn`, where *n* is the number of processors.

6 The Pace Cluster

6.1 Overview

This chapter is about the Pace Cluster which was built as part of this thesis. The chapter starts with a tutorial about how to add a new node to the Pace Cluster. It continues with the discussion of experimental runs. Runtimes and CPU utilization will be compared to a Linux Cluster. The chapter closes with a future outlook of the Pace Cluster. A list of all nodes can be found in the Appendix.

6.2 Adding a Node to the Pace Cluster

6.2.1 Required Files

To setup a new node for the Pace Cluster, the following items are required:

1. WMPI1.3 - The Windows Message Passing Interface version 1.3
2. PCG70P4 - a folder containing the PC GAMESS version 7.0, optimized for Pentium 4 processors
3. PCG70 - a folder containing the PC GAMESS version 7.0, for every processor type except Pentium 4
4. NAMD - a folder containing the NAMD + ...
5. The password to setup a new Windows user account

It is recommended to use the provided folders and files. If different versions are requested, the folders and files must be modified, and PC GAMESS must be configured for WMPI 1.3 usage. Additionally a work directory within the PCG folder must be created. If the prepared NAMD version is not desired, a way to run charmd.exe as service must be determined.

6.2.2 Creating a New User Account

The user account *pace* with a particular password must be on every node in the Pace Cluster. Consult the local system administrator to obtain the right password. To add a new user, hit the Windows Start button, select the Control Panel and click on User Accounts. Create a new account with the name *pace* and enter the password for the account.

IMPORTANT:

Make sure that a folder called *pace* is in the Documents and Settings folder. The following path is needed:

C:\Documents and Settings\pace

6.2.3 Install WMPI 1.3

Install WMPI 1.3 to the root folder C. It should have the path C:\WMPI1.3. Do not change the default settings during the installation. Now start the service and make sure that it is started automatically every time the machine is booted. Run the install_service batch file, found under C:\WMPI1.3\system\serviceNT\install_service.bat. Start the service by running C:\WMPI1.3\system\serviceNT\start_service.bat.

Right click on My Computer and select manage, like shown in Figure 6.1.

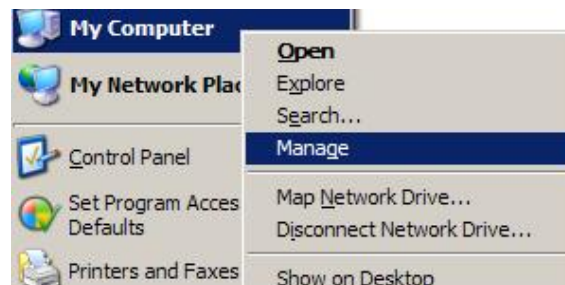


Figure 6.1 - Right click on My Computer

Select *Services* in Services and Applications, then double click on WMPI NT Service and set Startup type to *automatic*.

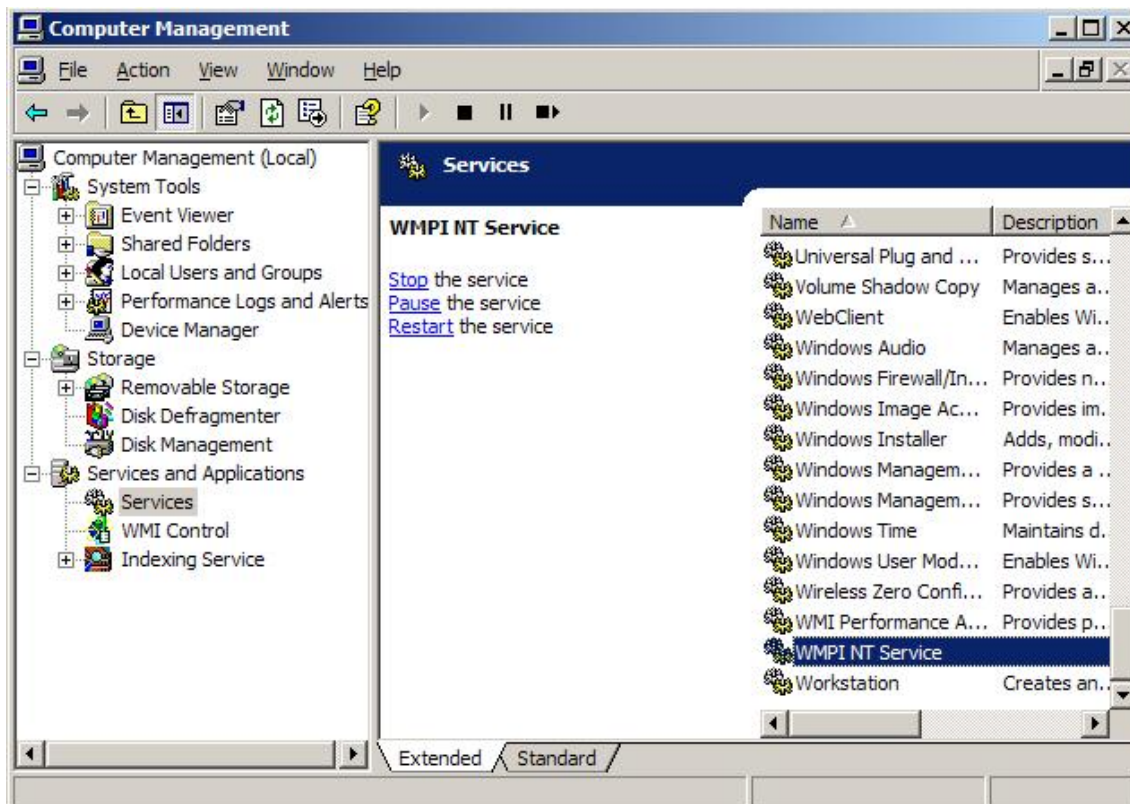


Figure 6.2 - WMPI NT Service

6.2.4 Install PC GAMESS

There are two versions of PC GAMESS; the regular one and an optimized one for Pentium 4 processors. The folder PCG70P4 contains the P4 version and the folder PCG70 contains the regular version. Copy the matching version to the local C root folder and rename it to C:\PCG.

6.2.5 Install NAMD

Copy the directory NAMD to the local C drive in the root folder C:\. The namd2.exe should now have the address C:\NAMD\namd2.exe. It is necessary to run the

executable charmd.exe as *service*. Only services run all the time, even if there no user is logged in. Charmd.exe is naturally not programmed as *service*, but the following work-around will fix the issue. The program XYNTService.exe can be started as *service* and can be configured to run other programs. The NAMD folder already includes a configured XYNTService version. Run the batch file C:\NAMD\install_service.bat.

6.2.6 Firewall Settings

Make sure that the following executables are not blocked by a firewall:

C:\WMPI1.3\system\serviceNT\wmpi_service.exe

C:\PCG\pcgamess.exe

C:\NAMD\namd2.exe

C:\NAMD\charmd.exe

If the Windows Firewall is used, click on Start, select Settings, Control Panel. Double click on the Windows Firewall icon, select the tab *Exceptions*.

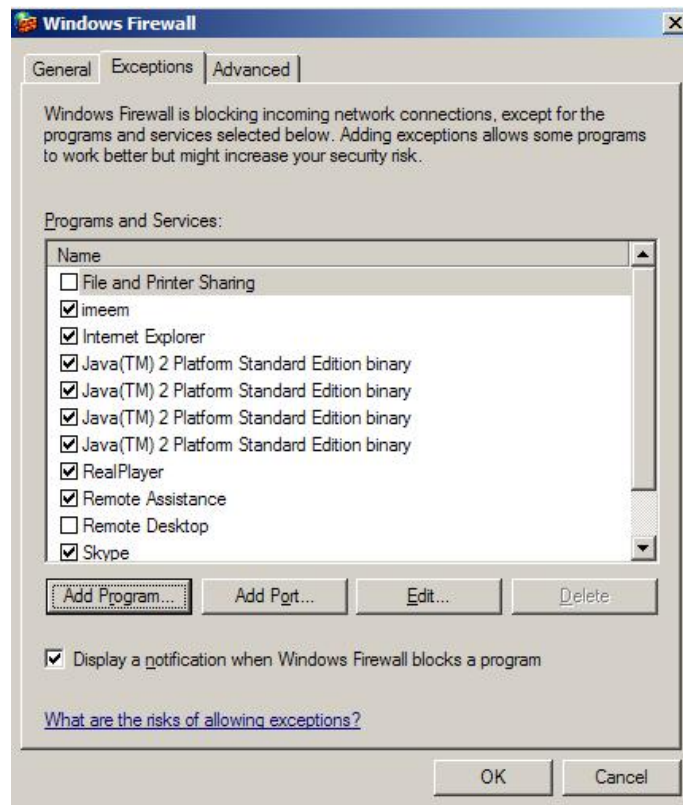


Figure 6.3 - Windows Firewall Configuration

Click on Add Program, like shown in the Figure above, then on Browse, select the above mentioned executables. The ping has to be enabled on every machine in the cluster. To enable it select the Advanced tab, click on ICMP settings and allow incoming echo requests, like shown in Figure 6.4.

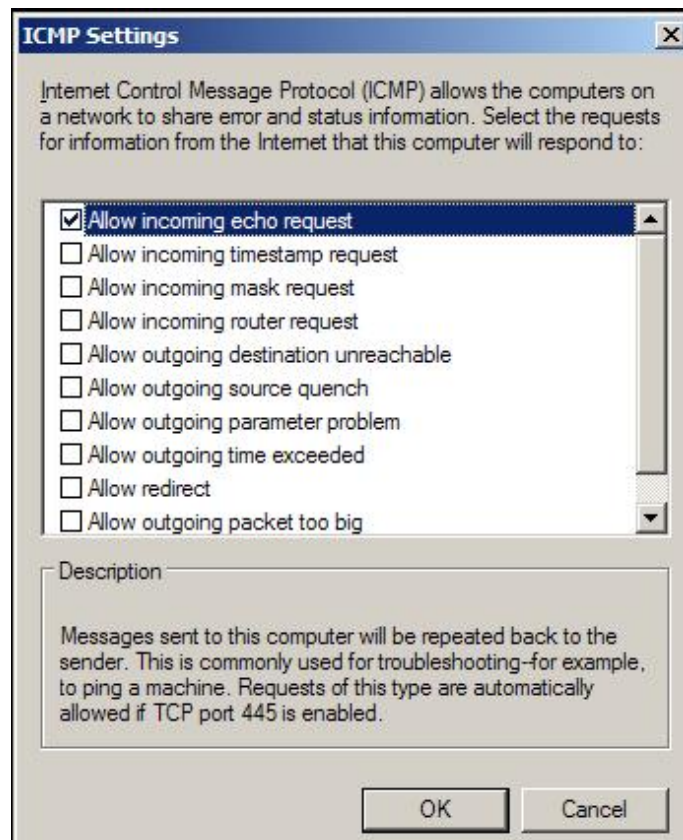


Figure 6.4 - ICMP Settings

If the Windows Firewall is not used, read the manual or contact the administrator.

6.2.7 Check the Services

Reboot the machine and check if the services `wmpi_service.exe`, `XYNTService.exe` and `charmd.exe` are running. Open the Task Manager by pressing `alt, ctrl and del`.

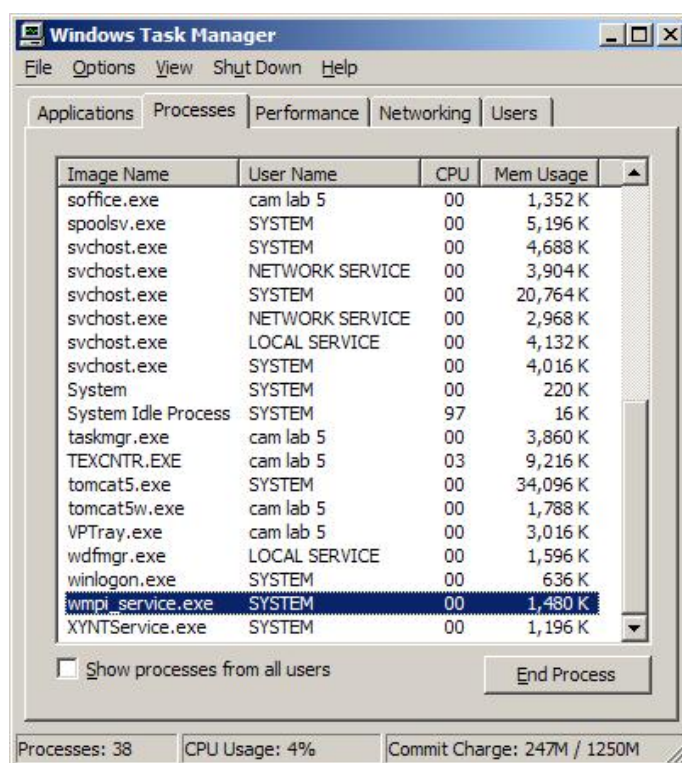


Figure 6.5 - Check Services

6.3 Diagram: Runtimes / Processors

The next diagram, shown in Figure 6.6, shows the runtimes of six calculations each performed with 1, 2, 4, 8, 16 and 32 processors. The six input files used for this run are shown in the Appendix (PC GAMESS Input files, B.1 - B.6). The calculations were run on the machines listed in Table 6.1

Table 6.1 - Run / Location of used Nodes

location	32 CPU _s	16 CPU _s	8 CPU _s	4 CPU _s	2 CPU _s	1 CPU
Cam Lab at 163	2	1	4	4	2	1
Tutor Lab at 163	0	0	4	0	0	0
Computer Lab, room B	30	15	0	0	0	0

Table 6.2 shows the runtime in seconds of every run:

Table 6.2 - Runtimes / Processors

Processors	Phenol	db7	db6 mp2	db5	Antracene	18cron6
1	958.8	4448.1	3093.7	822.4	5407.1	6439.2
2	493.5	2384.3	1511	436.4	2793.8	3184.5
4	261.2	1193.2	873.3	252.3	1527.8	1900.4
8	190.6	790.3	464.2	147.4	845.7	1261.8
16	169.8	452.9	332.9	107.6	521.8	828.6
32	152.7	353.3	230.7	80.9	342.6	591.7

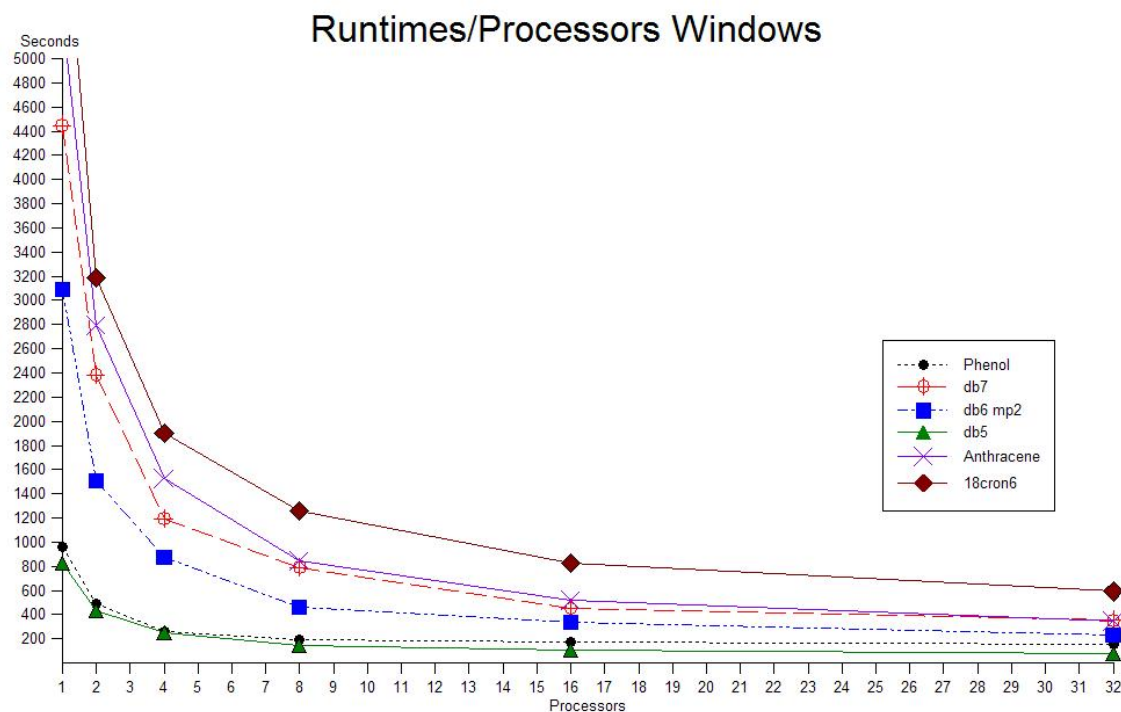


Figure 6.6 - Runtimes / Processors

The diagram and the table obviously show that the runtime decreases more slowly with more processors. While the difference from the 18cron6 run with one CPU and two CPUs is more than 3000 seconds, adding up to 32 CPUs from 16 CPUs, runtime gains less than 300 seconds. On the diagram the lines appear to converge. The next table and diagram show that with the doubling of nodes, the performance gain is less significant than the last one. However this is not the only reason for the apparent convergence of the lines. Even if the performance would increase 100% with every doubling of nodes, which would be the optimal case, the curve would look similar. As the nodes double, the length from one point to the next doubles on the x-axis and the height from one point to next halves on the y-axis. Over a long enough distance it would look as if the runtimes would meet, but the proportion between the values never change.

The table 6.3 shows the performance increase compared to the previous run, for every run, including the average run, for a certain number of processors. In this table, the performance increase augments every time the number of processors is doubled. After the initial run, where the number of CPUs increases from one CPU to two CPUs, the average performance of the cluster nearly doubles. The average performance increase of 80.9% the CPUs double from two to four. With every doubling of CPUs the performance increase is less than before. Adding doubling 16 machines to 32 only increases the average performance by about 34.7%.

Table 6.3 - Performance increase compared to the previous run

Processors	Phenol	db7	db6 mp2	db5	Antracene	18cron6	Average
2	94.2%	86.5%	104.7%	88.4%	93.5%	102.2%	94.9%
4	88.9%	99.8%	73.0%	74.5%	82.2%	67.5%	80.9%
8	37.0%	50.9%	88.1%	70.9%	80.0%	50.6%	62.9%
16	12.0%	74.4%	39.4%	36.9%	62.0%	52.2%	45.8%
32	11.1%	28.1%	44.2%	33.0%	52.3%	40.0%	34.7%

The figure 6.7 reflects the performance increase in relation to one CPU machine.

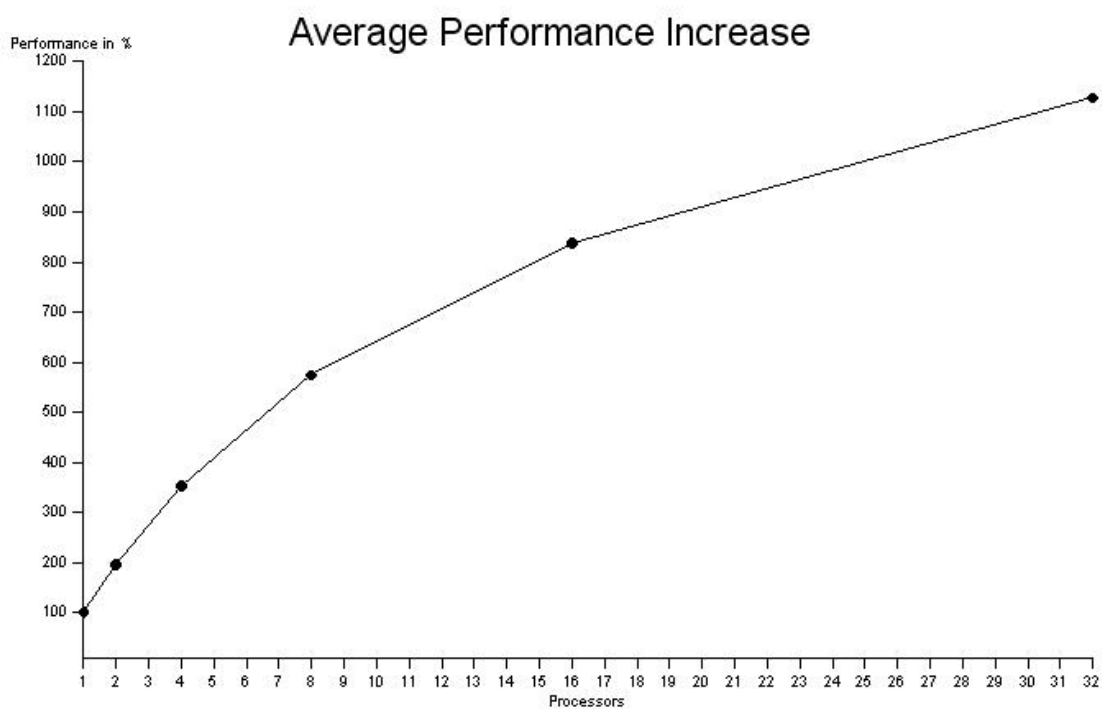


Figure 6.7 - Average Performance Increase

6.4 Diagram: Number of Basis Functions / CPU Utilization

The Number of Basis Functions / CPU Utilization diagram is based on the following data. The CPU Utilization in percentages was measured with 49 PC GAMESS calculations.

The calculations were run on the machines shown in table 6.1.

The table 6.4 should give an overview of the CPU utilization that was measured.

Table 6.4 - Number of Basis Functions / CPU Utilization

Calculation	Basis Functions	32 P	16 P	8 P	4 P	2 P
18crown6	568	59.69%	N/A	83.7%	97.32%	98.89%
anthracene	392	65.54%	69.93%	85.52%	98.02%	99.42%
benzene	180	41.52%	53.63%	75.81%	93.53%	97.66%
db1	74	19.74%	26.58%	47.44%	75.51%	93.65%
db2	134	33.26%	44.66%	68.55%	89.69%	97.13%
db3	194	43.31%	54.66%	76.21%	86.91%	97.97%
db4	254	48.88%	59.35%	79.2%	95.54%	98.69%
db5	314	50.92%	61.18%	80.09%	94.39%	98.84%
db6	374	59.99%	61.15%	82.46%	97.7%	98.84%
db7	434	61.33%	65.09%	81.64%	96.03%	98.39%
luciferin2	294	45.02%	58.39%	N/A	95.42%	98.64%
naphthalene	286	54.88%	64.08%	N/A	95.42%	99.11%
phenol	257	40.92%	63.4%	79.41%	94.54%	98.11%

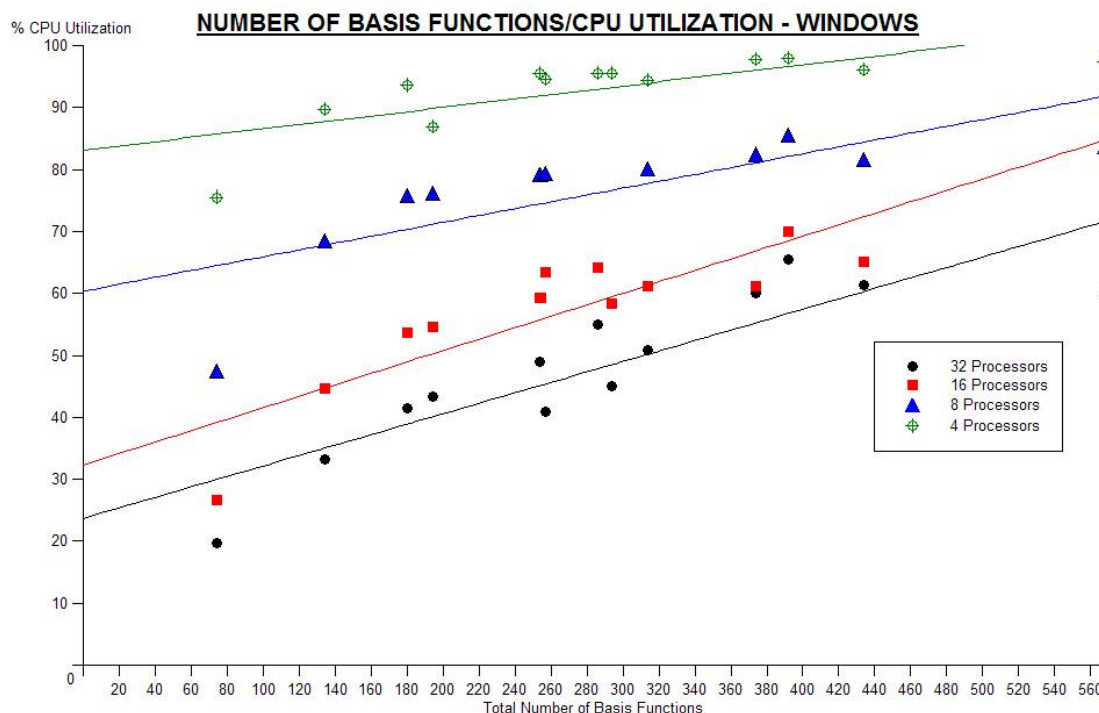


Figure 6.8 - Number of Basis Functions / CPU Utilization

The diagram shows that the runs with fewer CPUs have a better processor utilization than those run with more CPU utilization. Besides the normal communication overhead it should be noted that the 32 and 16 processors calculations used machines distributed over two different buildings and the 8 processor run used computers in two different rooms. It is also observable that with more basis functions the CPU utilization increases. According to the results of this experiment it is recommended to run small computations on fewer CPUs even if there are more available.

6.5 Network Topology and Performance

In this experiment, phenol mp2, as discussed in the Appendix (PC GAMESS Input files, B1), was run several times with 4 processors. For every run the composition of the machines was changed. The experiment shows how the global CPU time, wall

time, average CPU utilization per node and the total CPU utilization depends CPU power, network connections and what kind of role the composition of machines play.

1. Computers used in this run:

Table 6.5 - Computers Run 1

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
987.9 s	261.2 s	378.17%	94.54%

The first run gives an idea about the timing and utilization values for the four computers in the Cam Lab. These computers communicate over a 100 MBit full duplex connection. The data of the next runs will be compared to these values.

2. Computers used in this run:

Table 6.6 - Computers Run 2

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
Tutor Lab	3.2 GHz	1 GB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
932.3 s	261.2 s	357.01%	89.25%

In this run one computer was exchanged with a faster one in another room on the same floor. The communication between the three computers in the Cam Lab was still over a 100 MBit full duplex connection, but the way out of the Cam Lab was only 100 MBit half duplex. The computer in the tutor lab is more powerful, but the wall clock time did not change at all. It seems that the computer had to wait for the three slower ones, because the global CPU time is 50 seconds lower than at the first run and the average CPU utilization is lower.

3. Computers used in this run:

Table 6.7 - Computers Run 3

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Cam Lab	2.4 GHz	512 MB RAM	no
Tutor Lab	3.2 GHz	1 GB RAM	no
Tutor Lab	3.2 GHz	1 GB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
886.3 s	286.5 s	309.33%	77.33%

Another computer from the Cam Lab was replaced by a more powerful one from the Tutor Lab. The CPU utilization was again lower and the global CPU time decreased in comparison with the second run by about 50 seconds, but the wall clock time was 25 seconds more. One possible explanation for this result is congestion at the network during the time of the experiment.

4. Computers used in this run:

Table 6.8 - Computers Run 4

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Tutor Lab	3.2 GHz	1 GB RAM	no
Tutor Lab	3.2 GHz	1 GB RAM	no
Tutor Lab	3.2 GHz	1 GB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
825.0 s	264.3 s	312.16%	78.03%

The wall clock time is nearly similar to the first run, even though three ma-

chines were exchanged for more powerful ones with more memory. Later runs show that a slow head node will slow the cluster down.

5. Computers used in this run:

Table 6.9 - Computers Run 5

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
934.3 s	265.6 s	351.71%	87.93%

This run and the next two were very similar to the runs 2 to 4. The results of these runs were very similar and it seemed that communication between the buildings at 173 Wiliam St and One Pace Plaza did not play a role.

6. Computers used in this run:

Table 6.10 - Computers Run 6

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Cam Lab	2.4 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
884.1 s	263.1 s	336.06%	84.01%

7. Computers used in this run:

Table 6.11 - Computers Run 7

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
828.3 s	264.9 s	312.66%	78.16%

8. Computers used in this run:

Table 6.12 - Computers Run 8

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Tutor Lab	3.2 GHz	1 GB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
828.1 s	264.8 s	312.69%	78.34%

This and the next run show that the distribution for the four computers did not have a huge impact on the runtime behavior. The measured values of the four runs with the head node at the Cam Lab and the three slave nodes spread over One Pace Plaza and the Tutor Lab did not differ much from each other.

9. Computers used in this run:

Table 6.13 - Computers Run 9

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Tutor Lab	3.2 GHz	1 GB RAM	no
Tutor Lab	3.2 GHz	1 GB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
826.8 s	264.5 s	312.58%	78.14%

10. Computers used in this run:

Table 6.14 - Computers Run 10

location	CPU	RAM	master node
163 Wiliam St.	3.0 GHz	1 GB RAM	yes
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
937.4 s	261.8 s	358.09%	89.53%

This time the master node was changed to a more powerful CPU. The overall performance compared to the first run did not change. The master node was slowed down by its slaves. An indicator for this is the better global CPU time but the 5% smaller average CPU utilization.

11. Computers used in this run:

Table 6.15 - Computers Run 11

location	CPU	RAM	master node
One Pace Plaza	3.0 GHz	512 MB RAM	yes
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
780.0 s	216.0 s	361.07%	90.27%

This is the first run in which every CPU had 3.0 GHz. Every computer is equally powerful and they were all at the same physical location. This was also the first time a notable increase of speed was measured.

12. Computers used in this run:

Table 6.16 - Computers Run 12

location	CPU	RAM	master node
163 Wiliam St.	3.0 GHz	1 GB RAM	yes
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
780.6 s	217.7 s	358.51%	89.86%

The setting was similar to the previous one, but an equally powerful master node was located in a different building. The measured wall clock time differed by about 1.7 seconds and the CPU utilization was 0.43% better. It seems

that at least for small runs with four computers the performance loss of the communication between the network at 163 Wiliam St and One Pace Placa is negligible.

13. Computers used in this run:

Table 6.17 - Computers Run 13

location	CPU	RAM	master node
Cam Lab	1.8 GHz	512 MB RAM	yes
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
1117.0 s	448.9 s	248.86%	62.22%

The wall clock time of this and the following run demonstrated how a less powerful master node can slow down the whole system. In both cases the average node CPU utilization was only about 62%.

14. Computers used in this run:

Table 6.18 - Computers Run 14

location	CPU	RAM	master node
Cam Lab	1.8 GHz	512 MB RAM	yes
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
949.4 s	381.8 s	248.67%	62.18%

15. Computers used in this run:

Table 6.19 - Computers Run 15

location	CPU	RAM	master node
Cam Lab	2.4 GHz	512 MB RAM	yes
Cam Lab	1.8 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
Cam Lab	2.4 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
1101.0 s	376.7 s	292.30%	73.06%

The same computers were used as in run 14, but the master node was no longer the slowest machine in the cluster. It is observable that the master node was a critical component of a PC GAMESS cluster. Not using the 1.8 GHz machine as master saves 44 seconds wall clock time and 11% CPU utilization.

16. Computers used in this run:

Table 6.20 - Computers Run 16

location	CPU	RAM	master node
163 Wiliam St.	3.0 GHz	1 GB RAM	yes
Cam Lab	1.8 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
One Pace Plaza	3.0 GHz	512 MB RAM	no
glbl CPU time	wall clock time	total CPU util	node avrg CPU util
1101.0 s	376.7 s	292.30%	73.06%

This run was analog to the last one for 3.0 GHz CPUs.

The experiment has shown that a homogeneous cluster is much more powerful than a cluster consisting of different types of computers. Less powerful CPUs can slow down the faster ones and it is evident that the master node is a very critical component. The slave nodes spend a lot of time idling and waiting for the master.

6.6 Windows VS Linux

The next two diagrams demonstrate the performance differences between a Windows and a Linux cluster. The Linux Cluster consists of four machines with two processors and the Windows Cluster of 8 machines with single processors.

Table 6.21 - Computers of the Run: Cam Lab / Tutor Lab

Location	CPU	RAM	Master Node	Number of Computers
Cam Lab	2.4 GHz	512 MB RAM	yes	4
Tutor Lab	3.2 GHz	1 GB RAM	no	4

Table 6.22 - Computers of the Run: One Pace Plaza

Location	CPU	RAM	Master Node	Number of Computers
One Pace Plaza	3.0 GHz	512 MB RAM	yes	8

Each Linux computer that was used in this run has two CPUs.

Table 6.23 - Computers of both Linux runs

Location	CPU	RAM	Master Node	Number of Computers
Cam Lab	2 X 2 GHz	3.2 GB RAM	yes	4

Table 6.24 represents the CPU utilization that was measured during this experiment. The Linux version of PC GAMESS had a better CPU utilization, but the GAMESS version had better runtimes for these runs.

Table 6.24 - Basis Functions / CPU Utilization

Name	Basis Func	163 Wil.	OnePacePlz	Lin:PC GAMESS	Lin:GAMESS
18cron6	568	83.7%	90.05%	99.7%	97.09%
Anthracene	392	85.52%	92.93%	99.63%	95.87%
Benzene	180	75.81%	81.74%	99.36	95.73%
db1	74	47.44%	53.53	99.29%	85.9%
db2	134	68.55%	73.37	102.82%	88.06%
db3	194	76.21%	81.98	102.76%	63.09%
db4	254	79.2%	84.77	99.5%	86.68%
db5	314	80.09%	87.25	99.61%	90.98%
db6	374	82.46%	91.73	99.72%	92.98%
db7	434	81.64%	92.03	99.77%	93.85%
Luciferin2	257	79.41%	81.62%	99.64%	93.58%

Diagram 6.25 shows that the utilization of the Linux runs, was higher than the Windows runs. The Linux cluster consists of two processor machines. The Windows computers of single processors, which might have an impact at the communication between the nodes and the CPU utilization. The diagram also shows that the Linux PC GAMESS has a better CPU utilization for a smaller number of basis functions than the Linux GAMESS version. An explanation for the utilization difference between the Windows runs is the fact that different powerful computers were used at 163 Wiliam Street, which causes idle times, like previously pointed out.

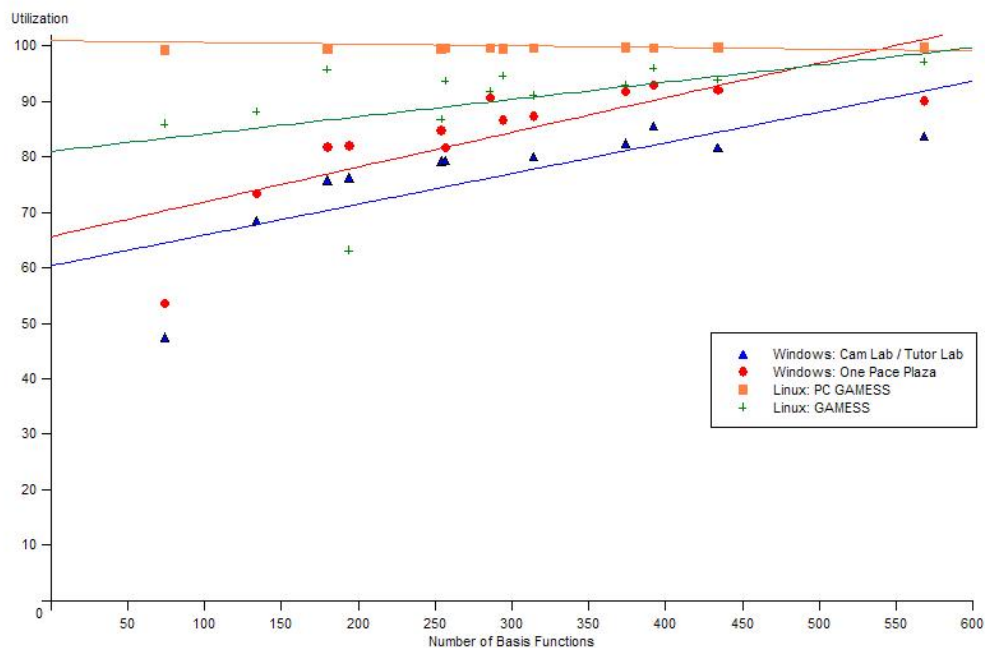


Figure 6.9 - CPU Utilization / Number of Basis Functions

Graph 6.10 shows the wall clock times of the runs db5, db6, db7 and 18cron6. The Windows Cluster consisting of the nodes at 163 Wiliams Street has the highest wall clock time, but also the slowest computers. The Linux cluster uses CPUs with 2 GHz while the Windows cluster at One Pace Plaza uses 3 GHz, but the Linux GAMESS cluster has similar run times and the Linux PC GAMESS version is just a bit slower. The Windows Cluster at 163 Wiliam Street has the worst wall clock time, even with more powerful nodes than the Linux Cluster. In this experiment the Linux cluster had better results than the Windows Cluster.

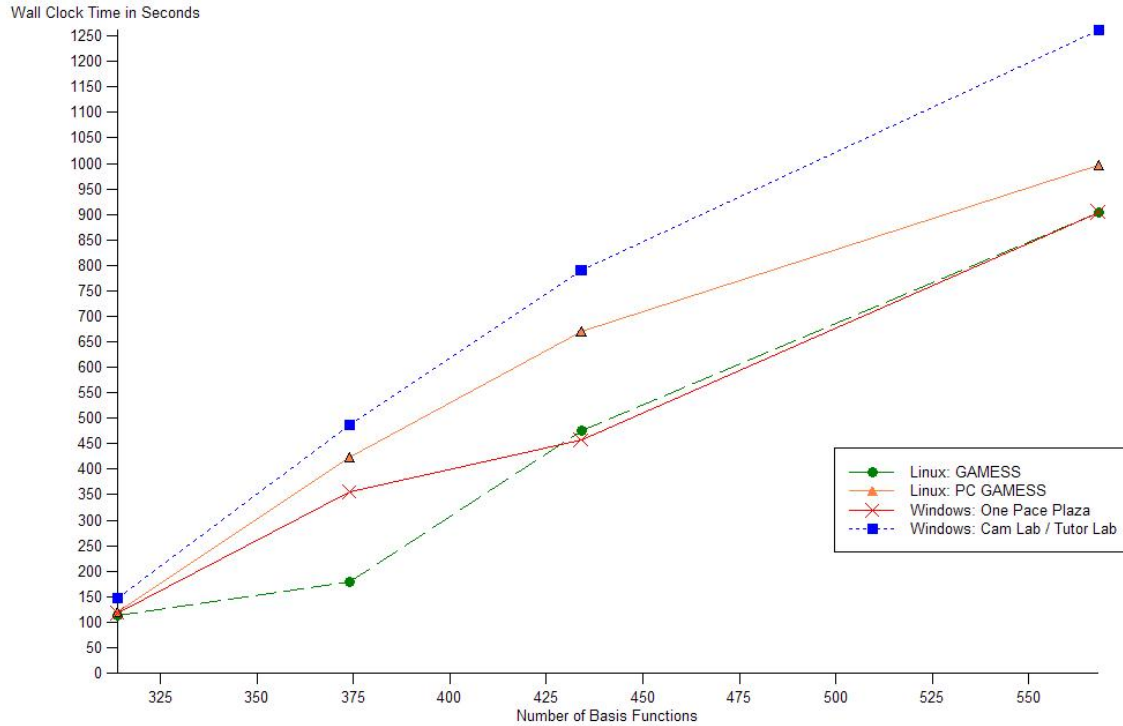


Figure 6.10 - Wall Clock Time / Number of Basis Functions

6.7 Conclusion of the Experiments

The experiments have shown that the total CPU utilization decreases by adding processors to the cluster. The performance increase for doubling the CPUs decreases and doubling 16 machines to 32 only increases the average performance by about 34.7%. The experiments have also shown that the communication between the building at One Pace Plaza and the one at 163 Wiliam Street does not significantly affect the performance. The choice of the master node has a big impact on the performance of the cluster. A slow master causes idle times of its more powerful slaves. The comparison showed that the Linux Cluster had the better CPU utilization and better run times. Furthermore it was demonstrated that the best clusters consist of equal powerful nodes at the same physical location.

6.8 Future Plans of the Pace Cluster

It is planned to add more nodes to the Pace Cluster. Computers from different rooms of the Computer Lab of One Pace Plaza will be added. There are theoretically 200 computers available at Pace University New York City campus. Further investigation and research will show, which computers are available and powerful enough to be added to the Pace Cluster. It is also planned to add computers from other campuses as well. The performance loss through the communication between the computers at One Pace Plaza and 163 William Street is minimal. Performance loss through communication or too much allocation of bandwidth are possible issues with a Cluster spread over different campuses. Future research will show if the communication between campuses will slow the cluster down or if the communication of the cluster produces so much congestion of the network that it interferes with other traffic of Pace University.

Besides the chemical calculations programs PC GAMESS and NAMD it is planned to install and run chemical visualization programs. One of the next steps will also be to run benchmarks for performance measurement like the LINPACK benchmark, which was introduced in a previous chapter.

7 The PC GAMESS Manager

7.1 Introduction to the PC GAMESS Manager

PC GAMESS is shipped without any GUI and is controlled over the Command Prompt, which is not very comfortable and user friendly. The idea of the PC GAMESS Manager is to allow the user to interact with PC GAMESS over an user friendly interface and to provide some convenient features to the user. It allows the user to create a queue of jobs and to execute them at given point of time. The PC GAMESS Manager checks the availability of nodes and allocates them dynamically. This is very useful, especially in a system like the Pace Cluster, with machines in many different physical locations and where the availability is not granted. The PC Gamess Manager has also an option to perform this availability check and to create a config file for NAMD.

There are other free managing tools like RUNpcg or Webmo which will be introduced at the end of this chapter.

7.2 The PC GAMESS Manager User's Manual

7.2.1 Installation

The PC GAMESS Manager runs on the master node of the cluster where the initial PC GAMESS process is started. The PC GAMESS Manager was programmed in the language C#. Copy the PC GAMESS Manager folder to the local hard disc and run the setup.exe. Like all C# programs it needs the Microsoft .NET Framework 2.0. The install routine of the PC GAMESS Manager will check for it automatically and ask if download and installation is desired. PC GAMESS should be installed as described in the chapter *The Pace Cluster*. Every machine in the

cluster should be entered in a file called pclist.txt. It should be available over the path C:\PCG\pclist.txt and contains only the host name or IP addresses of the machines separated by line breaks.

This would be an example for a proper pclist.txt:

```
pace-cam-01  
pace-cam-02  
pace-cam-03  
pace-cam-04  
172.20.102.62  
172.20.102.214  
172.20.103.119  
172.20.103.112
```

7.2.2 The First Steps

To start the program, execute the *PC GAMESS Manager.application*. After the program was launched the GUI will look like the following picture.

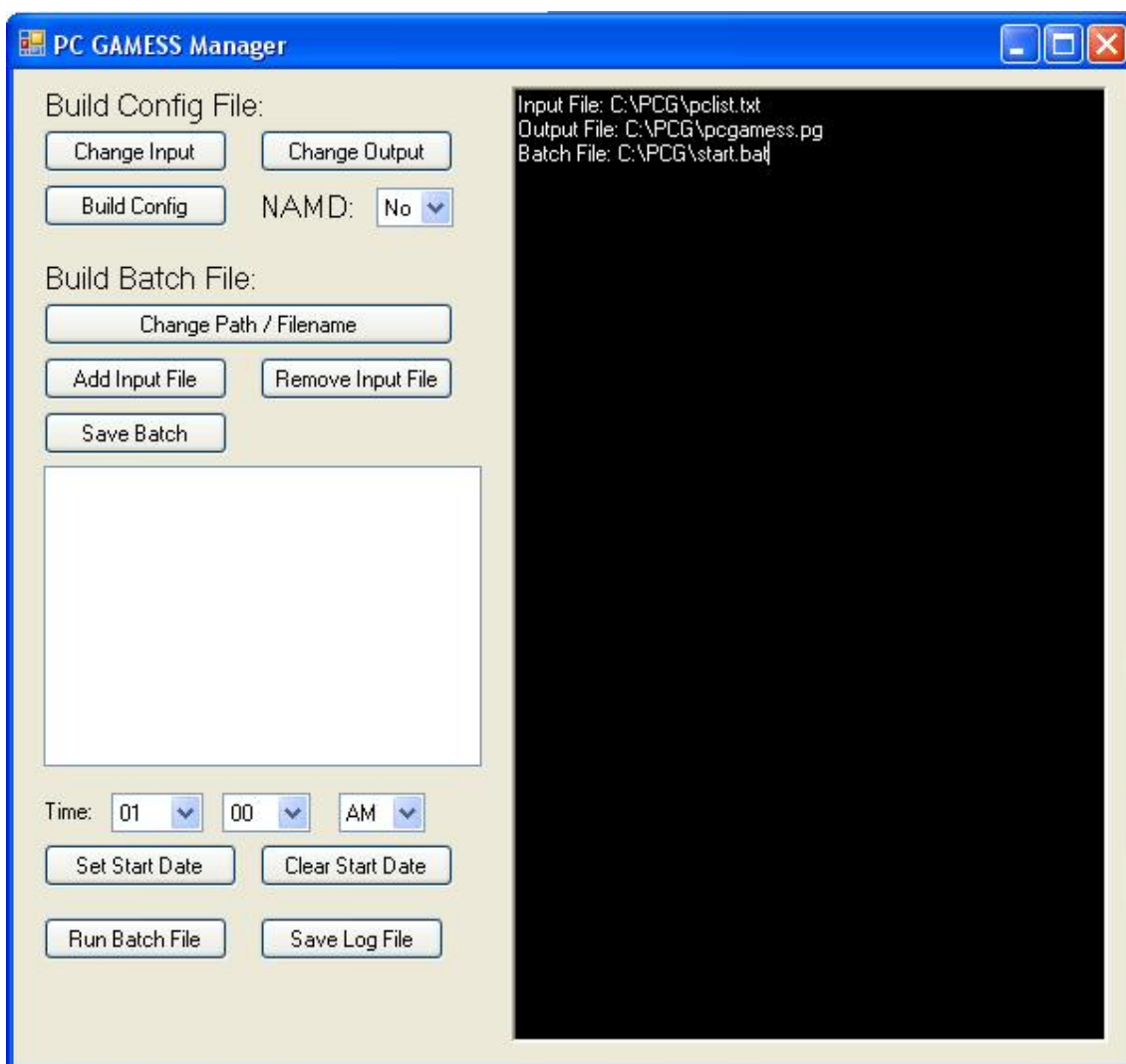


Figure 7.1 - GUI of the PC GAMESS Manager

On the left side of the GUI you see the control panel and on the right side you see the output shell of the program. The output shell will confirm every successful executed command or will give the according error message. The whole process to run a PC Gamess program is separated into three steps: building a config file, building a batch file and to run the batch file.

7.2.3 Building a Config File

First build a PC GAMESS config file which is also called progroup file, which was described in the chapter WMPI. It includes the list of nodes on which the next PC GAMESS program should be executed. After the Build Config button is pressed the following steps will be automatically executed:

A file with a list of machines is read, which can include host names as well as IP addresses. Every machine on the list is pinged. If the machine is available it will be added to the progroup file. The default configuration uses C:\PCG\pclist.txt as input file and writes the proucgroup file to C:\PCG\pcgameess.pg. The path of both files can be changed by using the button *Change Input* and *Change Output*. The output shell will give more detailed information about the result of every ping command. The machines will only be added if the ping was successful. If the message that the DNS lookup was successful is displayed, then it is possible that the ping command was blocked by a firewall.

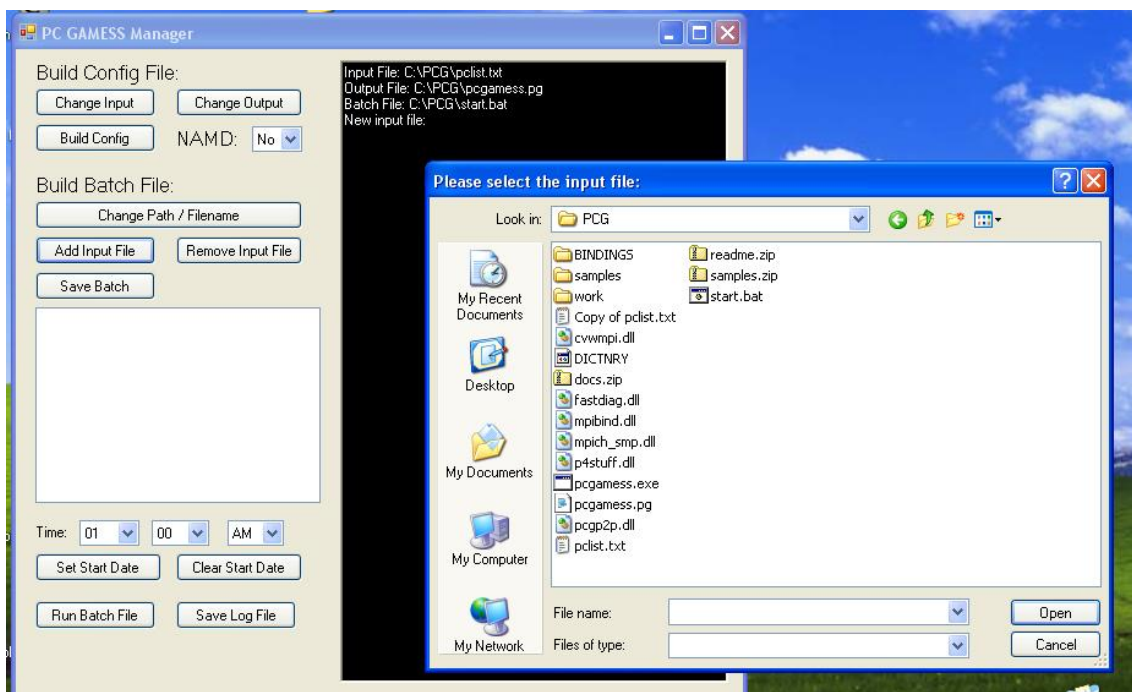


Figure 7.2 - Change the Input File of the PC GAMESS Manager

7.2.4 Building a NAMD Nodelist File

The PC GAMESS Manager gives the option to create a NAMD Nodelist file. Switch the option in the dropdown box NAMD to *yes*. The default input file can stay the same, because NAMD and PC GAMESS are using the same machines. Change the output file, select the target file to overwrite or create a new one. Click on Build Config and the availability of the machines is checked and a standard NAMD nodefile will be created.

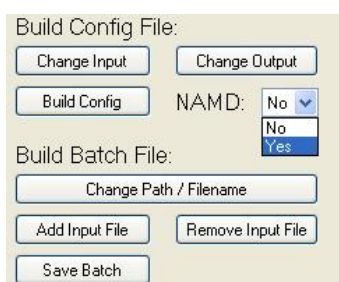


Figure 7.3 - Build Config File Menu

7.2.5 Building a Batch File

PC GAMESS allows the user to put more then one job in a queue. To add a job click on Add Input File and select it. The output shell should confirm the selection and it should appear in the list box like shown at the following picture:

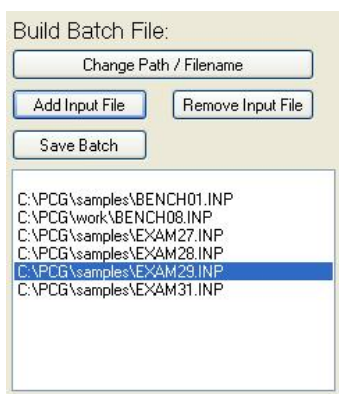


Figure 7.4 - Build Batch File Menu

An input file from the selection can be removed by selecting it from the list and clicking on Remove Input File. With the option Change Path / Filename the default path of the batch file C:\PCG\start.bat can be altered. If you created a list of jobs click on Save Batch.

7.2.6 Run the Batch File

The batch file can be run immediately by clicking on Run Batch File or set the timer to run it later.

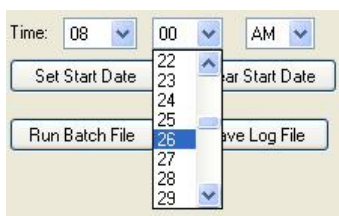


Figure 7.5 - Run Batch File Menu

To use the timer select the point of time and hit Set Start Time. The timer with the Clear Start Time button can be cleared. The timer options is very useful to run huge calculations over night in the computer pools of Pace University while the computers can be used during the day by students. When the batch file starts a windows command prompts pops up and shows the status.

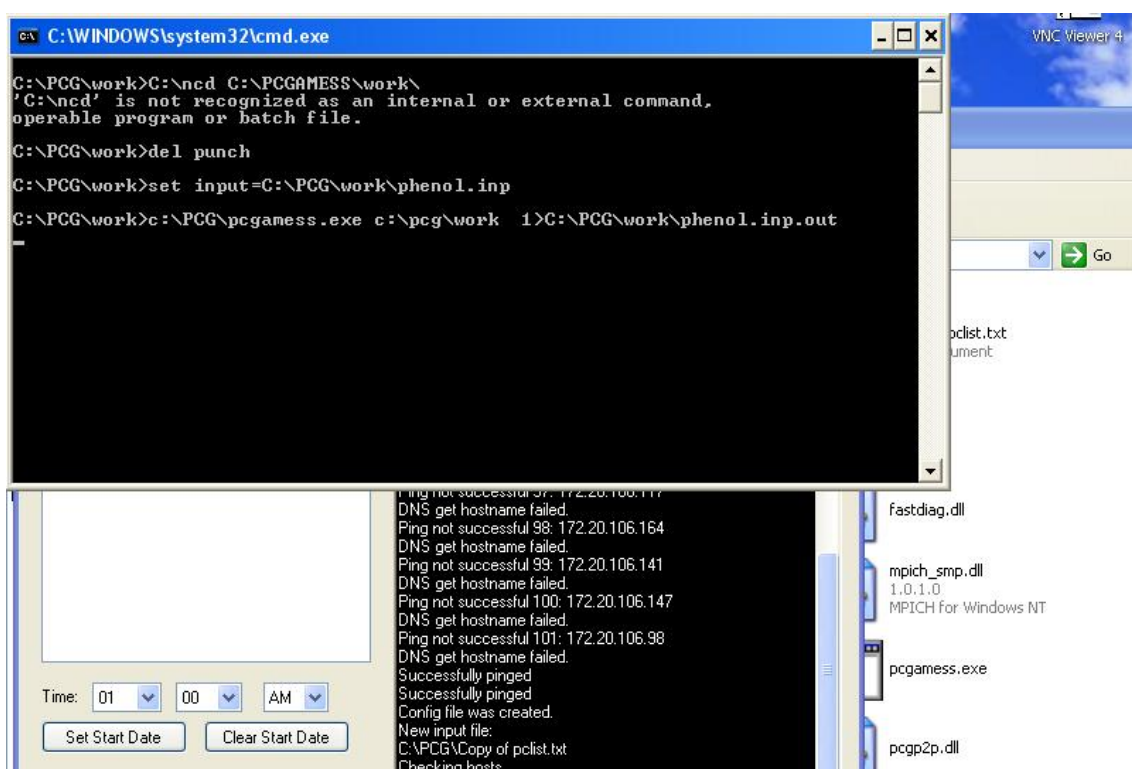


Figure 7.6 - Running PC GAMESS with the PC GAMESS Manager

When all jobs are finished the output shell will print the needed time for the whole job queue. The output files of PC GAMESS are written in the directory as the input files. The PC GAMESS Manager just adds .out to the file name of the input files.

7.2.7 Save Log File

The Save Log File command saves the current output of the output shell in a log file. The log file is created in the C:\PCG folder. It has a unique time stamp as name, every time the button is pressed a new log file is created.

7.3 RUNpcg

RUNpcg[27] was published in July 2003. It couples PC GAMESS with other free software that is available via internet. RUNpcg enables the user to build molecules

and to compose PC GAMESS input files and to view the structure of the output file by using the free software.



Figure 7.7 - Menu and Runscript RUNpcg

There are many free programs used to built, and draw the molecules, for example ArgusLab4[28], ChemSketch5[29] or ISIS/Draw6[30] as well as commercial software like HyperChem 8[31] or PCModel 9[32].

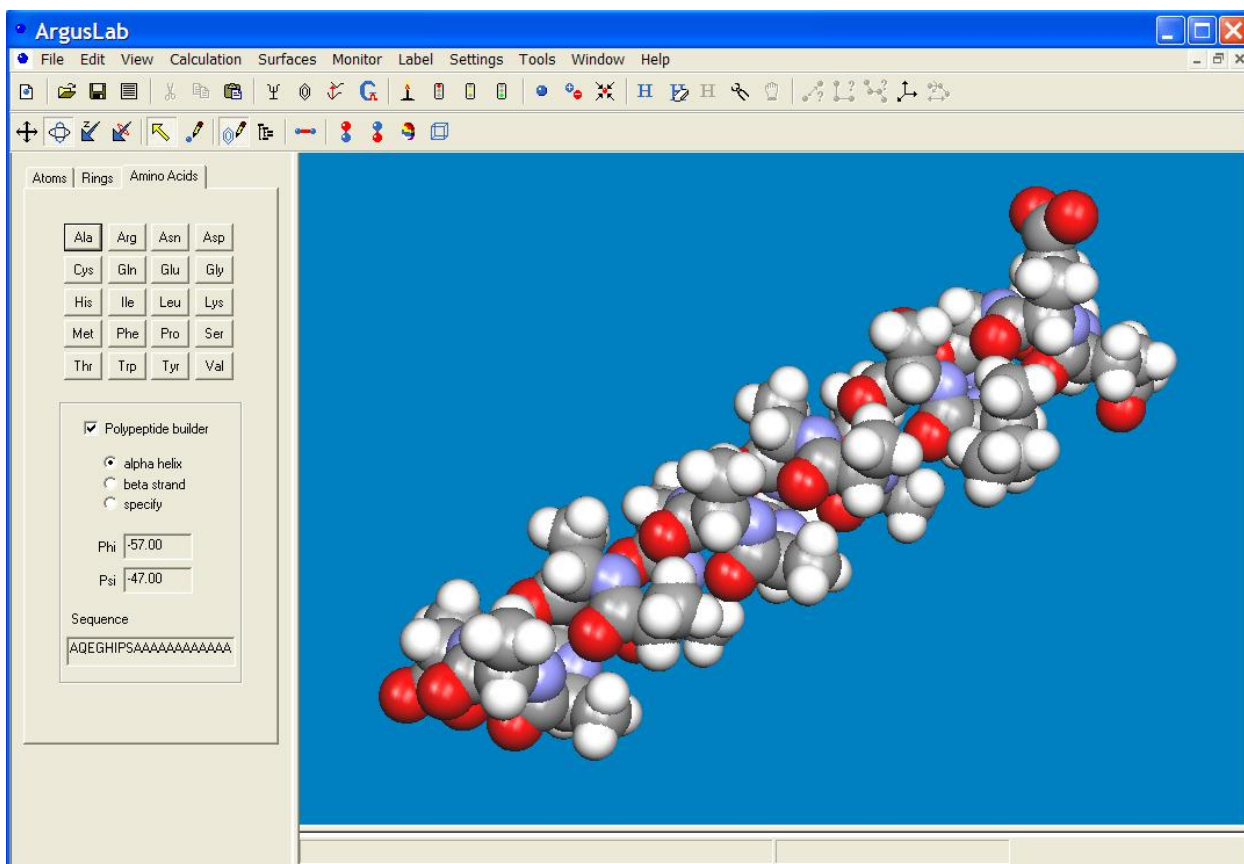


Figure 7.8 - ArgusLab4

Different programs can be used to build the input file for RUNpcg, like gOpenMol10[33], VMD11[34], RasWin12[35], Molekel13[36], Molden17[37] and ChemCraft15[39] to create a graphical representation of the output file.

7.4 WebMo

WebMo[38] runs on a Linux/Unix system and is accessed via web-browser. It is not necessary that the browser runs on the same computer, WebMo can be used over a network. In addition to the free version there is WebMo Pro, a commercial version with some extra features. WebMo comes with a 3D Java based molecular editor.

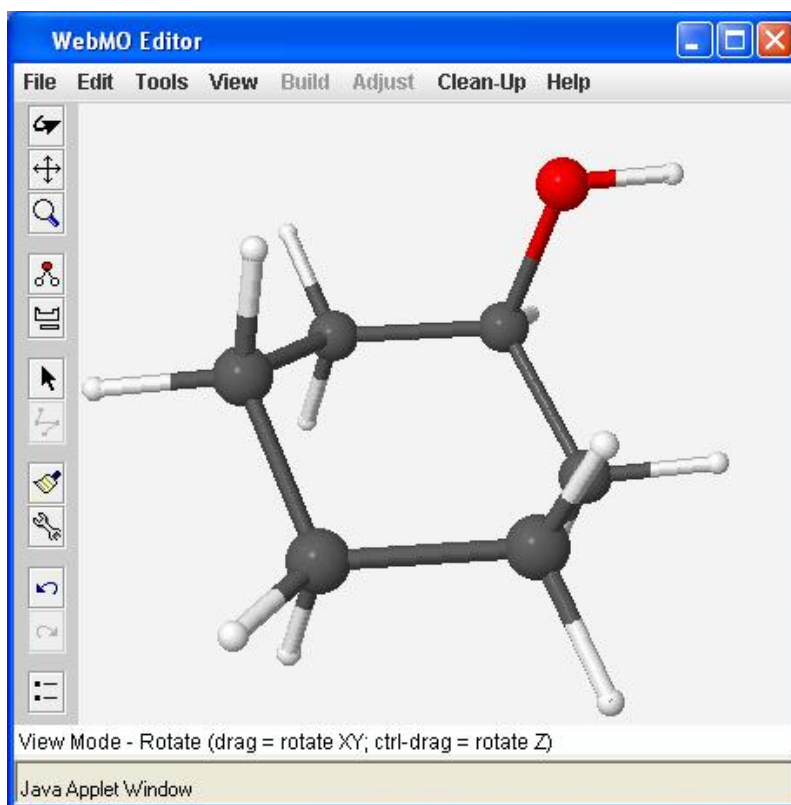


Figure 7.9 - WebMo 3D Molecular Editor

The editor has true 3D rotation, zooming, translation and the ability to adjust bond distances, angles and dihedral angles.

WebMo has features like a job manager, which allows the user to monitor and to control jobs. The job options allows the user to edit the Gaussian input file before it is computed. WebMo offers different options to view the result. It has a 3D viewer which allows the user to rotate and zoom in the visualization. Beside the raw text output WebMo gives the option to view the result in tables of energies, rotational constants, partial charges, bond orders, vibrational frequencies, and NMR shifts.

7.5 RUNpcg, WebMo and the PC GAMESS Manager

RUNpcg and WebMo focus clearly on the visualization of the input and output files in rotatable 3D graphics and offer additionally job managers to monitor and edit

the queue. The PC GAMESS Manager does not offer graphical features and has a statical queue without interaction possibility. The PC GAMESS Manager was customized for the Pace Cluster and it was built to address its main problems like starting jobs at a certain point of time, checking if the nodes are online and building config as well as batch files.

8 Conclusion

This thesis demonstrates how to build an Ad-Hoc Windows Cluster to perform high performance computing in an inexpensive way. It was shown how to establish a connection between the nodes with the free communication software WMPI 1.3 and how to use PC GAMESS and NAMD for scientific computing. The free available XYNT Service was used to run charmd as a service and allows the user to use the computers is no user is logged in. Besides the free software the cluster uses the network infrastructure of Pace University and common office computers in the computer pools spread over the campus.

The PC GAMESS Manager was developed as part of this thesis to provide the users with an user friendly interface. The PC GAMESS Manager can be used to create a list of currently available computers at Pace University and to create PC GAMESS and NAMD config files. A comfortable job queue and a timer provides the user with the ability to put jobs in a queue and to start them at a desired point of time.

The experimental runs have shown that the physical locations of the nodes at the New York City Campus does not have a huge impact on the performance of the cluster. The experiments have also shown that small computations run better on fewer nodes, because the timer overhead for loading the program and setting up the cluster takes more time with more nodes. But for large computations, which take hours or days, the time to set up the cluster is negligible and more nodes will pay off. The experiments also demonstrate the importance to use equal powerful machines. Less powerful nodes will slow down the whole cluster system. Especially the master node is a very critical point and a slow master will cause idle times for its slaves. For this reasons it is recommended to start the calculations from a

computer in a pool via remote administration tools. To use a computer in the pool would also have the advantage of limited network traffic to the pool and the computations would not interfere with the bandwidth between buildings of Pace University.

It is planned to add further nodes to the Pace Cluster in the future. There are 200 computers at One Pace Plaza, which will be added over the next months. The plan also includes adding computers from other campuses as well in the case that there will be no bandwidth or performance issues. Besides running PC GAMESS and NAMD it is planned to run additionally visualization programs and benchmarks like LINPACK for a better measure of the performance.

A Node List of the Pace Cluster

A.1 Cam Lab

There are four Windows Nodes in the Cam Lab at 168 Wiliam Street, Pace University New York City Campus.

Host Name	CPU	RAM
pace-cam-01	2.4 GHz	512 MB
pace-cam-02	2.4 GHz	512 MB
pace-cam-03	2.4 GHz	512 MB
pace-cam-04	2.4 GHz	512 MB

A.2 Tutor Lab

There are six Windows Nodes in the Tutor Lab at 168 Wiliam Street, Pace University New York City Campus.

Host Name	CPU	RAM
E315-WS5	3.2 GHz	1 GB
E315-WS6	3.2 GHz	1 GB
E315-WS7	3.2 GHz	1 GB
E315-WS32	3.2 GHz	1 GB
E315-WS2	3.2 GHz	1 GB
E315-WS3	3.2 GHz	1 GB

A.3 Computer Lab - Room B

There are thirty Windows Nodes in the Computer Lab at One Pace Plaza in room B, Pace University New York City Campus.

Physical Name	IP Address	CPU	RAM
PC 72	172.20.102.62	3 GHz	512 MB
PC 73	172.20.102.214	3 GHz	512 MB
PC 74	172.20.103.119	3 GHz	512 MB
PC 75	172.20.103.112	3 GHz	512 MB
PC 76	172.20.103.110	3 GHz	512 MB
PC 77	172.20.103.111	3 GHz	512 MB
PC 78	172.20.101.129	3 GHz	512 MB
PC 79	172.20.100.184	3 GHz	512 MB
PC 80	172.20.104.212	3 GHz	512 MB
PC 81	172.20.105.237	3 GHz	512 MB
PC 82	172.20.103.162	3 GHz	512 MB
PC 83	172.20.100.165	3 GHz	512 MB
PC 84	172.20.105.243	3 GHz	512 MB
PC 85	172.20.100.10	3 GHz	512 MB
PC 86	172.20.102.242	3 GHz	512 MB
PC 87	172.20.106.39	3 GHz	512 MB
PC 88	172.20.106.43	3 GHz	512 MB
PC 89	172.20.106.75	3 GHz	512 MB
PC 90	172.20.106.70	3 GHz	512 MB
PC 91	172.20.106.38	3 GHz	512 MB
PC 92	172.20.106.49	3 GHz	512 MB
PC 93	172.20.106.58	3 GHz	512 MB

Physical Name	IP Address	CPU	RAM
PC 94	172.20.106.170	3 GHz	512 MB
PC 95	172.20.106.124	3 GHz	512 MB
PC 96	172.20.106.108	3 GHz	512 MB
PC 97	172.20.106.117	3 GHz	512 MB
PC 98	172.20.106.164	3 GHz	512 MB
PC 99	172.20.106.141	3 GHz	512 MB
PC 100	172.20.106.147	3 GHz	512 MB
PC 101	172.20.106.98	3 GHz	512 MB

B PC GAMESS Inputfiles

B.1 Phenol

```
$CONTRL SCFTYP=RHF MPLEVL=2 RUNTYP=ENERGY
ICHARG=0 MULT=1 COORD=ZMTMPC $END
$SYSTEM MWORDS=50 $END
$BASIS GBASIS=N311 NGAUSS=6 NDFUNC=2 NPFUNC=2 DIFFSP=.TRUE.
$END
$SCF DIRSCF=.TRUE. $END
$DATA
C6H6O
C1 1
C 0.0000000 0 0.0000000 0 0.0000000 0 0 0 0
C 1.3993653 1 0.0000000 0 0.0000000 0 1 0 0
C 1.3995811 1 117.83895 1 0.0000000 0 2 1 0
C 1.3964278 1 121.36885 1 0.0310962 1 3 2 1
C 1.3955209 1 119.96641 1 -0.0350654 1 4 3 2
C 1.3963050 1 121.35467 1 0.0016380 1 1 2 3
H 1.1031034 1 120.04751 1 179.97338 1 6 1 2
H 1.1031540 1 120.24477 1 -179.97307 1 5 4 3
H 1.1031812 1 120.04175 1 179.97097 1 4 3 2
H 1.1027556 1 119.23726 1 -179.97638 1 3 2 1
O 1.3590256 1 120.75481 1 179.99261 1 2 1 6
H 0.9712431 1 107.51421 1 -0.0155649 1 11 2 1
H 1.1028894 1 119.31422 1 179.99642 1 1 2 3
$END
```

B.2 db7

```
$CONTRL SCFTYP=RHF DFTTYP=B3LYP5 runtyp=energy $END
$SYSTEM MEMORY=3000000 $END
$SCF DIRSCF=.TRUE. $END
$BASIS GBASIS=n311 ngauss=6 NDFUNC=1 NPFUNC=1 DIFFSP=.TRUE.
DIFFSP=.TRUE. $END
$GUESS GUESS=HUCKEL $END
$DATA
Seven double Bonds
C1
C 6.0 0.18400 0.00000 1.01900
C 6.0 1.29600 0.00000 1.77000
H 1.0 -0.79500 0.00000 1.49500
H 1.0 2.27400 0.00000 1.29300
```

```

C 6.0 1.26800 0.00000 3.21600
C 6.0 2.38000 0.00000 3.96700
H 1.0 0.28900 0.00000 3.69300
H 1.0 3.35800 0.00000 3.49000
C 6.0 2.35100 0.00000 5.41300
C 6.0 3.46300 0.00000 6.16400
H 1.0 1.37300 0.00000 5.89000
H 1.0 4.44200 0.00000 5.68700
C 6.0 3.43500 0.00000 7.61000
C 6.0 4.54700 0.00000 8.36100
H 1.0 2.45700 0.00000 8.08700
H 1.0 5.52600 0.00000 7.88500
C 6.0 4.51900 0.00000 9.80700
C 6.0 5.63100 0.00000 10.55800
H 1.0 3.54100 0.00000 10.28400
H 1.0 6.61000 0.00000 10.08200
C 6.0 5.60200 0.00000 12.00200
C 6.0 6.70900 0.00000 12.75300
H 1.0 4.63100 0.00000 12.49300
H 1.0 7.70400 0.00000 12.31900
H 1.0 6.63900 0.00000 13.83700
C 6.0 0.21300 0.00000 -0.42500
C 6.0 -0.89400 0.00000 -1.17600
H 1.0 1.18400 0.00000 -0.91500
H 1.0 -1.88900 0.00000 -0.74100
H 1.0 -0.82400 0.00000 -2.25900
$END

```

B.3 db6 mp2

```

$CONTRL SCFTYP=RHF MPLEVL=2 runtyp=energy $END
$SYSTEM MEMORY=3000000 $END
$SCF DIRSCF=.TRUE. $END
$BASIS GBASIS=n311 ngauss=6 NDFUNC=1 NPFUNC=1 DIFFSP=.TRUE.
DIFFS=.TRUE. $END
$GUESS GUESS=HUCKEL $END
$DATA
Six Double Bonds
C1
H 1.0 0.15900 0.00000 -0.00900
C 6.0 0.10500 0.00000 1.07600
C 6.0 1.22400 0.00000 1.81000
H 1.0 -0.88300 0.00000 1.52500

```

```

H 1.0 2.18700 0.00000 1.30500
C 6.0 1.21600 0.00000 3.25400
C 6.0 2.34000 0.00000 3.98800
H 1.0 0.24500 0.00000 3.74500
H 1.0 3.31000 0.00000 3.49600
C 6.0 2.33300 0.00000 5.43500
C 6.0 3.45700 0.00000 6.16800
H 1.0 1.36200 0.00000 5.92600
H 1.0 4.42800 0.00000 5.67700
C 6.0 3.45000 0.00000 7.61500
C 6.0 4.57400 0.00000 8.34900
H 1.0 2.47900 0.00000 8.10700
H 1.0 5.54500 0.00000 7.85800
C 6.0 4.56700 0.00000 9.79500
C 6.0 5.69000 0.00000 10.52900
H 1.0 3.59600 0.00000 10.28700
H 1.0 6.66200 0.00000 10.03900
C 6.0 5.68300 0.00000 11.97300
C 6.0 6.80200 0.00000 12.70800
H 1.0 4.71900 0.00000 12.47900
H 1.0 7.79000 0.00000 12.25800
H 1.0 6.74800 0.00000 13.79200
$END

```

B.4 db5

```

$CONTRL SCFTYP=RHF DFTTYP=B3LYP5 runtyp=energy $END
$SYSTEM MEMORY=3000000 $END
$SCF DIRSCF=.TRUE. $END
$BASIS GBASIS=n311 ngauss=6 NDFUNC=1 NPFUNC=1 DIFFSP=.TRUE.
DIFFS=.TRUE. $END
$GUESS GUESS=HUCKEL $END
$DATA
Five double bonds
C1
H 1.0 0.07000 0.00000 0.02600
C 6.0 0.03300 0.00000 1.11100
C 6.0 1.16200 0.00000 1.82900
H 1.0 -0.94800 0.00000 1.57500
H 1.0 2.11800 0.00000 1.30900
C 6.0 1.17600 0.00000 3.27300
C 6.0 2.31000 0.00000 3.99000
H 1.0 0.21200 0.00000 3.77800

```

```

H 1.0 3.27400 0.00000 3.48400
C 6.0 2.32600 0.00000 5.43600
C 6.0 3.46000 0.00000 6.15300
H 1.0 1.36200 0.00000 5.94200
H 1.0 4.42300 0.00000 5.64800
C 6.0 3.47500 0.00000 7.60000
C 6.0 4.60900 0.00000 8.31700
H 1.0 2.51200 0.00000 8.10600
H 1.0 5.57300 0.00000 7.81200
C 6.0 4.62300 0.00000 9.76100
C 6.0 5.75300 0.00000 10.47900
H 1.0 3.66700 0.00000 10.28000
H 1.0 6.73400 0.00000 10.01400
H 1.0 5.71500 0.00000 11.56400
$END

```

B.5 Anthracene

```

$CONTRL SCFTYP=RHF DFTTYP=B3LYP5 runtyp=energy $END
$SYSTEM MEMORY=3000000 $END
$SCF DIRSCF=.TRUE. $END
$BASIS GBASIS=n311 ngauss=6 NDFUNC=1 NPFUNC=1 DIFFSP=.TRUE.
DIFFS=.TRUE. $END
$GUESS GUESS=HUCKEL $END
$DATA
anthracene
C1
H 1.0 0.00000 0.00000 -0.01000
C 6.0 0.00000 0.00000 1.08600
H 1.0 -0.00100 2.15000 1.22500
C 6.0 0.00000 1.20900 1.78600
C 6.0 0.00100 -1.20700 3.18200
C 6.0 0.00000 1.21600 3.18700
C 6.0 0.00000 -1.20900 1.78400
C 6.0 0.00300 0.00300 3.88700
C 6.0 0.00100 2.42400 3.89400
H 1.0 -0.00100 -2.15900 1.23600
H 1.0 0.00500 -0.93800 5.83500
H 1.0 0.00200 -2.16400 3.71600
C 6.0 0.00100 2.43200 5.29400
H 1.0 -0.00100 3.37300 3.34600
C 6.0 -0.00100 3.64200 5.99900
C 6.0 0.00400 1.21900 5.99400

```

```

H 1.0 0.01200 0.28500 7.95600
C 6.0 0.00300 0.01100 5.28700
C 6.0 0.00200 3.64400 7.39700
H 1.0 -0.00500 4.59900 5.46500
H 1.0 -0.00100 4.59400 7.94500
C 6.0 0.00700 2.43500 8.09500
H 1.0 0.01000 2.43500 9.19100
C 6.0 0.00800 1.22600 7.39500
$END

```

B.6 18crown6

```

$CONTRL SCFTYP=RHF DFTTYP=B3LYP5 runtyp=optimize $END
$SYSTEM MEMORY=3000000 $END
$SCF DIRSCF=.TRUE. $END
$BASIS GBASIS=n311 ngauss=6 NDFUNC=1 NPFUNC=1 DIFFSP=.TRUE.
DIFFS=.TRUE. $END
$GUESS GUESS=HUCKEL $END
$DATA
18-crown-6
C1
O 8.0 1.82200 0.01200 -2.14000
O 8.0 2.32200 -1.59900 0.12600
O 8.0 0.19700 -2.01000 1.98500
C 6.0 -1.89100 -1.17100 2.89500
C 6.0 3.11900 -0.51400 -1.88900
C 6.0 2.93800 -1.82500 -1.15000
C 6.0 2.13800 -2.79500 0.87000
C 6.0 1.52600 -2.45500 2.18400
C 6.0 -0.46300 -1.65500 3.20500
H 1.0 -2.38300 -1.02700 3.70500
H 1.0 -2.33200 -1.86200 2.39300
H 1.0 3.59600 -0.63700 -2.71200
H 1.0 2.38700 -2.40600 -1.68000
H 1.0 3.79000 -2.24800 -1.01900
H 1.0 2.98100 -3.23600 0.99700
H 1.0 1.55200 -3.38800 0.39300
H 1.0 1.53300 -3.21800 2.76500
H 1.0 2.03500 -1.75200 2.59700
H 1.0 0.01200 -0.93000 3.62000
H 1.0 -0.47000 -2.40000 3.80900
O 8.0 -1.82200 -0.01200 2.14000
O 8.0 -2.32200 1.59900 -0.12600

```

O 8.0 -0.19700 2.01000 -1.98500
C 6.0 1.89100 1.17100 -2.89500
C 6.0 -3.11900 0.51400 1.88900
C 6.0 -2.93800 1.82500 1.15000
C 6.0 -2.13800 2.79500 -0.87000
C 6.0 -1.52600 2.45500 -2.18400
C 6.0 0.46300 1.65500 -3.20500
H 1.0 2.38300 1.02700 -3.70500
H 1.0 2.33200 1.86200 -2.39300
H 1.0 -3.59600 0.63700 2.71200
H 1.0 -2.38700 2.40600 1.68000
H 1.0 -3.79000 2.24800 1.01900
H 1.0 -2.98100 3.23600 -0.99700
H 1.0 -1.55200 3.38800 -0.39300
H 1.0 -1.53300 3.21800 -2.76500
H 1.0 -2.03500 1.75200 -2.59700
H 1.0 -0.01200 0.93000 -3.62000
H 1.0 0.47000 2.40000 -3.80900
\$END

References

- [1] Lightning,
http://www.lanl.gov/news/index.php?fuseaction=home.story&story_id=1473
- [2] Los Alamos,
<http://www.lanl.gov/projects/asci/>
- [3] World Community Grid,
<http://www.worldcommunitygrid.org/>
- [4] Ian Foster,
<http://www-fp.mcs.anl.gov/~foster/>
- [5] Ian Foster's Grid Definition,
<http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf>
- [6] IBM's Grid Definition,
http://www-304.ibm.com/jct09002c/isv/marketing/emerging/grid_wp.pdf
- [7] CERN's Grid Definition,
<http://gridcafe.web.cern.ch/gridcafe/whatisgrid/whatis.html>
- [8] Robert W. Lucke Building Clustered Linux Systems,
Page 22, 1.6 Revisiting the Definition of Cluster
- [9] Hongzhang Shan, Jaswinder Pal Singh, Leonid Oliker, Rupak Biswas,
<http://crd.lbl.gov/~oliker/papers/ipdps01.pdf>
- [10] LINPACK,
<http://www.netlib.org/benchmark/hpl/>
- [11] Top500,
<http://www.top500.org/>
- [12] Interview with Ian Foster,
http://www.betanews.com/article/print/Interview_The_Future_in_Grid_Computing/1109004118
- [13] Sun aims to sell computing like books, tickets, zdnet,
http://news.zdnet.com/2100-9584_22-5559559.html
- [14] PlayStation 3 Cell chip aims high, zdnet,
http://news.zdnet.com/2100-9584_22-5563803.html
- [15] Moore's Law,
<http://www.intel.com/technology/mooreslaw/index.htm>

- [16] The MPI Forum,
www.mpi-forum.org
- [17] WMPI II,
<http://www.criticalsoftware.com/hpc/>
- [18] Ethernet,
<http://www.ethermanage.com/ethernet/10gig.html>
- [19] Infiniband,
<http://www.intel.com/technology/infiniband/>
- [20] Myrinet,
<http://www.myri.com/myrinet/overview/>
- [21] WMPI,
http://parallel.ru/ftp/mpi/wmpi/WMPI_EuroPVMMPI98.pdf
- [22] RFC 1014 - XDR: External Data Representation standard,
<http://www.faqs.org/rfcs/rfc1014.html>
- [23] PC GAMESS,
<http://classic.chem.msu.su/gran/gamess/>
- [24] GAMESS (US),
<http://www.msg.ameslab.gov/GAMESS/>
- [25] SMP Definition,
http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci214218,00.html
- [26] NAMD,
<http://www.ks.uiuc.edu/Research/namd/>
- [27] RUNpcg,
<http://chemsoft.ch/qc/Manualp.htm#Intro>
- [28] ArgusLab,
<http://www.planaria-software.com/>
- [29] ACD/ChemSketch Freeware,
<http://www.acdlabs.com/download/chemsk.html>
- [30] ISIS/Draw,
<http://www.mdli.com>
- [31] HyperChem,
<http://www.hyper.com/>

- [32] PCModel,
<http://serenasoft.com/index.html>
- [33] gOpenMol is maintained by Leif Laaksonen, Center for Scientific Computing,
Espoo, Finland.
<http://www.csc.fi/gopenmol/>
- [34] VMD,
<http://www.ks.uiuc.edu>
- [35] RasWin,
<http://www.umass.edu/microbio/rasmol/getras.htm>
- [36] Molekel,
<http://www.cscs.ch/molekel>
- [37] Molden,
<http://www.cmbi.ru.nl/molden/molden.html>
- [38] WebMo,
Webmo.net
- [39] ChemCraft,
<http://www.chemcraftprog.com/>